WELCOME coupon available
Use code WELCOME at checkout through July 31, 2026.
WELCOME View sales
×

How To Write FiveM Scripts Using AI: Complete Developer&#…

JULY 4, 2026 / LUA SCRIPTING

Writing FiveM scripts has traditionally required deep knowledge of Lua, JavaScript, and the FiveM API. Today, AI tools like Claude Code, GitHub Copilot, and ChatGPT are change how developers create everything from ESX scripts to complex standalone systems. This comprehensive guide shows you exactly how to leverage AI for FiveM development, with concrete examples and proven

workflows.

Why AI-Assisted FiveM Development Changes Everything

Traditional FiveM script development requires mastering multiple technologies simultaneously: Lua for server-side logic, JavaScript for NUI interfaces, SQL for database operations, and the extensive FiveM native functions library. AI tools compress months of learning into hours of productive coding.

Awesome Tools for FiveM Development

Real Impact for Server Owners:

- Reduce custom script development time by 60-80%
- Generate boilerplate code instantly for common patterns
- Debug complex synchronization issues between client and server
- Convert ideas directly into working prototypes

Essential AI Tools for FiveM Development

Claude Code (Anthropic)

Claude Code excels at FiveM development because it understands context across multiple files and can generate complete resource structures. Install via command

line:

```
pip install claude-code
claude-code init --project fivem-resource
```

FiveM-Specific Advantages:

- Generates complete fxmanifest.lua configurations
- Understands ESX, QBCore, and VRP frameworks
- Creates matching client/server event handlers automatically

GitHub Copilot

Integrates directly into VS Code, providing real-time suggestions as you type. Particularly effective for:

- Completing native function calls
- Generating event handler patterns
- Auto-completing database queries

ChatGPT with Custom Instructions

Configure ChatGPT specifically for FiveM by setting custom instructions:

```
You are a FiveM script developer. Always use:
- Lua 5.4 syntax for server scripts
- Modern JavaScript for client scripts
- FiveM natives from the latest game build
- Proper event security with server-side validation
```

Step-by-Step: Creating Your First

AI-Generated Script

Let's build a complete vehicle shop script using Claude Code, demonstrating the entire workflow from concept to deployment.

Step 1: Define Clear Requirements

Create a `requirements.md` file:

```
## Vehicle Shop Script Requirements
- Framework: ESX Legacy 1.9.0+
- Database: MySQL with oxmysql
- Features:
 - Browse vehicles by category
 - Test drive system with time limit
 - Finance options with weekly payments
 - Admin commands for adding vehicles
- UI: Modern NUI with React
```

Step 2: Generate the Resource Structure

Command to Claude Code:

```
claude-code generate "Create a complete FiveM
vehicle shop resource based on requirements.md"
```

This generates:

```
vehicle_shop/
├── fxmanifest.lua
├── config.lua
├── server/
│ ├── main.lua
```

```
│   ├── database.lua
│   └── commands.lua
├── client/
│   ├── main.lua
│   ├── showroom.lua
│   └── testdrive.lua
├── ui/
│   ├── index.html
│   ├── style.css
│   └── script.js
└── sql/
    └── vehicle_shop.sql
```

Step 3: AI-Generated fxmanifest.lua

```lua
fx_version 'cerulean'
game 'gta5'

author 'YourName'
description 'AI-Generated Vehicle Shop'
version '1.0.0'

shared_scripts {
  '@es_extended/imports.lua',
  '@ox_lib/init.lua',
  'config.lua'
}

client_scripts {
  'client/*.lua'
}

server_scripts {
  '@oxmysql/lib/MySQL.lua',
  'server/*.lua'
}

```

```
ui_page 'ui/index.html'

files {
 'ui/index.html',
 'ui/style.css',
 'ui/script.js'
}

dependencies {
 'es_extended',
 'oxmysql',
 'ox_lib'
}
```

Step 4: Implement Core Functionality with AI

Prompt for Server-Side Vehicle Loading:

```
-- AI-Generated: server/database.lua
local vehicles = {}

CreateThread(function()
 local result = MySQL.query.await('SELECT * FROM vehicle_shop WHERE stock > 0')

 for i = 1, #result do
 local vehicle = result[i]
 vehicles[vehicle.model] = {
 model = vehicle.model,
 name = vehicle.name,
 price = vehicle.price,
 category = vehicle.category,
 stock = vehicle.stock
 }
 end
```

```
17.
18.   print(('[^2INFO^7] Loaded %d vehicles from
    database'):format(#result))
19.  end)
20.
21.  -- Secure server event for purchasing
22.
    RegisterNetEvent('vehicle_shop:server:purchaseVehicle
    function(vehicleModel, paymentMethod)
23.   local src = source
24.   local xPlayer = ESX.GetPlayerFromId(src)
25.
26.   -- Security check
27.   if not vehicles[vehicleModel] then
28.   return TriggerClientEvent('ox_lib:notify', src, {
29.   title = 'Error',
30.   description = 'Invalid vehicle model',
31.   type = 'error'
32.   })
33.   end
34.
35.   local vehicleData = vehicles[vehicleModel]
36.
37.   -- Payment processing
38.   if paymentMethod == 'cash' then
39.   if xPlayer.getMoney() >= vehicleData.price then
40.   xPlayer.removeMoney(vehicleData.price)
41.   -- Generate plate and spawn vehicle
42.   local plate = GeneratePlate()
43.
44.   MySQL.insert('INSERT INTO owned_vehicles (owner,
    plate, vehicle) VALUES (?, ?, ?)', {
45.   xPlayer.identifier,
46.   plate,
47.   json.encode({model = vehicleModel, plate = plate})
48.   })
49.
50.
    TriggerClientEvent('vehicle_shop:client:spawnVehicle',
```

```
    src, vehicleModel, plate)
51.   else
52.   TriggerClientEvent('ox_lib:notify', src, {
53.   title = 'Insufficient Funds',
54.   description = ('You need
    $%d'):format(vehicleData.price),
55.   type = 'error'
56.   })
57.   end
58.   end
59.  end)
```

Step 5: Client-Side Implementation

AI Prompt: “Create client-side showroom with camera controls and vehicle preview”

```
1.  -- AI-Generated: client/showroom.lua
2.  local currentVehicle = nil
3.  local cam = nil
4.  local inShowroom = false
5.
6.  local showroomCoords = vector3(-42.34, -1097.35,
    26.42)
7.  local vehicleSpawnCoords = vector4(-47.52, -1097.29,
    26.42, 340.0)
8.
9.  function OpenShowroom(category)
10.   inShowroom = true
11.   DoScreenFadeOut(500)
12.   Wait(500)
13.
14.   -- Setup camera
15.   cam = CreateCam('DEFAULT_SCRIPTED_CAMERA', true)
16.   SetCamCoord(cam, -54.31, -1097.29, 28.42)
17.   PointCamAtCoord(cam, vehicleSpawnCoords.x,
    vehicleSpawnCoords.y, vehicleSpawnCoords.z)
```

```
  SetCamActive(cam, true)
  RenderScriptCams(true, false, 0, true, false)

  -- Request vehicles from server
ESX.TriggerServerCallback('vehicle_shop:getVehicles', function(vehicles)
  SendNUIMessage({
  action = 'showVehicles',
  vehicles = vehicles,
  category = category
  })
  SetNuiFocus(true, true)
  end, category)

  DoScreenFadeIn(500)
 end

 RegisterNUICallback('previewVehicle', function(data, cb)
  if currentVehicle then
  DeleteEntity(currentVehicle)
  end

  local model = GetHashKey(data.model)
  RequestModel(model)

  while not HasModelLoaded(model) do
  Wait(10)
  end

  currentVehicle = CreateVehicle(model, vehicleSpawnCoords.x, vehicleSpawnCoords.y,
  vehicleSpawnCoords.z, vehicleSpawnCoords.w, false, false)
  SetEntityAsMissionEntity(currentVehicle, true, true)
  SetVehicleOnGroundProperly(currentVehicle)
  SetVehicleDoorsLocked(currentVehicle, 2)
```

```
51.
52.     cb('ok')
53.   end)
```

Advanced AI Techniques for Complex Systems

Multi-File Context Management

When developing interconnected systems like phone scripts, provide AI with complete context:

```
1.   # Feed multiple files to Claude Code
2.   claude-code analyze --files
    "server/*.lua,client/*.lua,config.lua"
3.     --prompt "Add a garage system that integrates with
    the existing vehicle shop"
```

Framework-Specific Prompting

Different frameworks require different approaches. Here's how to optimize AI prompts for each:

For ESX:

```
1.   Generate an ESX job script with:
2.   - Grades/ranks system using ESX.Jobs
3.   - Salary payments via ESX.GetJobs()
4.   - Boss menu with society funds
5.   - Use esx_society patterns
```

For QBCore:

```
Create a QBCore gang system with:
- Gang territories using PolyZone
- Reputation system with QB.Shared.Gangs
- Stash/storage using qb-inventory
- Gang management via qb-gangmenu patterns
```

For VRP Scripts:

```
Build a VRP business system using:
- vRP.getUserId for player identification
- vRP permissions system
- vRP money methods (giveMoney, tryPayment)
- vRPclient tunnel for client callbacks
```

Database Schema Generation

AI excels at creating optimized database structures:

```
-- AI Prompt: "Create MySQL schema for advanced vehicle shop with financing"
-- Generated Result:

CREATE TABLE IF NOT EXISTS `vehicle_shop_catalog` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `model` varchar(60) NOT NULL,
  `name` varchar(60) NOT NULL,
  `manufacturer` varchar(60) DEFAULT NULL,
  `category` varchar(60) DEFAULT NULL,
  `price` int(11) NOT NULL,
  `stock` int(11) DEFAULT 10,
  `max_speed` int(11) DEFAULT NULL,
  `acceleration` float DEFAULT NULL,
  `braking` float DEFAULT NULL,
  `handling` float DEFAULT NULL,
```

```
  PRIMARY KEY (`id`),
  UNIQUE KEY `model` (`model`),
  KEY `idx_category` (`category`),
  KEY `idx_price` (`price`)
 ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

 CREATE TABLE IF NOT EXISTS `vehicle_financing` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `identifier` varchar(60) NOT NULL,
  `vehicle_id` int(11) NOT NULL,
  `down_payment` int(11) NOT NULL,
  `weekly_payment` int(11) NOT NULL,
  `weeks_remaining` int(11) NOT NULL,
  `last_payment` timestamp DEFAULT CURRENT_TIMESTAMP,
  `repo_warning_sent` boolean DEFAULT FALSE,
  PRIMARY KEY (`id`),
  KEY `idx_identifier` (`identifier`),
  FOREIGN KEY (`vehicle_id`) REFERENCES
`owned_vehicles`(`id`) ON DELETE CASCADE
 ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

Debugging and Optimization with AI

Performance Profiling Prompts

```
 -- Prompt: "Add performance monitoring to this
resource"
 -- AI generates:

 local performanceStats = {
  events = {},
  queries = {}
 }

```

```lua
local function profileEvent(eventName, func)
 return function(...)
  local startTime = GetGameTimer()
  local result = {func(...)}
  local executionTime = GetGameTimer() - startTime

  performanceStats.events[eventName] = performanceStats.events[eventName] or {}
  table.insert(performanceStats.events[eventName], executionTime)

  if executionTime > 50 then
  print(('[^3WARNING^7] Event %s took %dms'):format(eventName, executionTime))
  end

  return table.unpack(result)
  end
 end

 -- Wrap existing events
RegisterNetEvent('vehicle_shop:server:purchaseVehicle', profileEvent('purchaseVehicle', purchaseVehicleHandler))
```

Security Validation

AI can identify and fix security vulnerabilities:

```lua
 -- Prompt: "Add security validation to this client-to-server event"
 -- Before (vulnerable):
 RegisterNetEvent('giveWeapon', function(weapon)
  local src = source
  GiveWeaponToPed(GetPlayerPed(src), weapon, 250, false, true)
```

```
end)

-- After (AI-secured):
local allowedWeapons = {
 ['WEAPON_PISTOL'] = true,
 ['WEAPON_COMBATPISTOL'] = true
}

RegisterNetEvent('giveWeapon', function(weapon)
 local src = source
 local xPlayer = ESX.GetPlayerFromId(src)

 -- Multiple validation layers
 if not xPlayer then return end
 if not allowedWeapons[weapon] then
 return DropPlayer(src, 'Attempted to spawn invalid weapon')
 end
 if not xPlayer.hasPermission('weapons.spawn') then
 return TriggerClientEvent('ox_lib:notify', src, {
 title = 'Access Denied',
 type = 'error'
 })
 end

 -- Log the action
 print(('[^3WEAPON^7] %s spawned %s'):format(xPlayer.getName(), weapon))

 xPlayer.addWeapon(weapon, 250)
end)
```

NUI Development with AI

Creating modern interfaces for FiveM requires HTML/CSS/JavaScript expertise. AI streamlines this process:

React Component Generation

```
// Prompt: "Create a React vehicle shop UI with categ
filtering"
// AI generates:

import React, { useState, useEffect } from 'react';
import './VehicleShop.css';

const VehicleShop = () => {
  const [vehicles, setVehicles] = useState([]);
  const [selectedCategory, setSelectedCategory] =
useState('all');
  const [selectedVehicle, setSelectedVehicle] =
useState(null);
  const [searchTerm, setSearchTerm] = useState('');

  useEffect(() => {
  window.addEventListener('message', handleMessage);
  return () => window.removeEventListener('message',
handleMessage);
  }, []);

  const handleMessage = (event) => {
  const { action, data } = event.data;

  switch(action) {
  case 'showVehicles':
  setVehicles(data.vehicles);
  break;
  case 'updateStock':
  updateVehicleStock(data.model, data.stock);
  break;
  }
  };

  const filteredVehicles = vehicles.filter(vehicle =>
  const matchesCategory = selectedCategory === 'all' |
```

```
    vehicle.category === selectedCategory;
    const matchesSearch = vehicle.name.toLowerCase()
    .includes(searchTerm.toLowerCase());
    return matchesCategory && matchesSearch;
    });

    const purchaseVehicle = (vehicle, paymentMethod) =>
fetch(`https://${GetParentResourceName()}/purchaseVeh
{
    method: 'POST',
    headers: { 'Content-Type': 'application/json' },
    body: JSON.stringify({
    model: vehicle.model,
    paymentMethod: paymentMethod
    })
    });
    };

    return (
    <div className="vehicle-shop-container">
    <div className="shop-header">
    <h1>Premium Deluxe Motorsport</h1>
    <input
    type="text"
    placeholder="Search vehicles..."
    value={searchTerm}
    onChange={(e) => setSearchTerm(e.target.value)}
    className="search-input"
    />
    </div>

    <div className="category-filters">
    {['all', 'sports', 'super', 'sedan', 'suv',
'motorcycle'].map(cat => (
    <button
    key={cat}
    className={`category-btn ${selectedCategory === cat
'active' : ''}`}
```

```
    onClick={() => setSelectedCategory(cat)}
    >
    {cat.toUpperCase()}
    </button>
    ))}
    </div>

    <div className="vehicles-grid">
    {filteredVehicles.map(vehicle => (
    <VehicleCard
    key={vehicle.model}
    vehicle={vehicle}
    onSelect={() => setSelectedVehicle(vehicle)}
    />
    ))}
    </div>

    {selectedVehicle && (
    <VehicleDetails
    vehicle={selectedVehicle}
    onPurchase={purchaseVehicle}
    onClose={() => setSelectedVehicle(null)}
    />
    )}
    </div>
    );
  };
```

Common AI Pitfalls and Solutions

Problem 1: Outdated Native Functions

Issue: AI suggests deprecated FiveM natives **Solution:** Always specify the game build in your prompt:

```
1. Use only FiveM natives compatible with game build 3095+
2. Replace deprecated natives like GetPlayerPed(-1) with PlayerPedId()
```

Problem 2: Memory Leaks in Loops

Issue: AI creates infinite loops without proper cleanup **Solution:** Request garbage collection patterns:

```
1.  -- AI-improved version with cleanup
2.  local activeThreads = {}
3.
4.  local function createManagedThread(id, func)
5.   if activeThreads[id] then
6.   -- Kill existing thread
7.   activeThreads[id] = false
8.   end
9.
10.   activeThreads[id] = true
11.
12.   CreateThread(function()
13.   while activeThreads[id] do
14.   func()
15.   Wait(0)
16.   end
17.   end)
18.  end
19.
20.  -- Cleanup on resource stop
21.  AddEventHandler('onResourceStop', function(resourceName)
22.   if GetCurrentResourceName() == resourceName then
23.   for id, _ in pairs(activeThreads) do
24.   activeThreads[id] = false
25.   end
26.   end
```

```
end)
```

Problem 3: Synchronization Issues

Issue: Client-server state mismatches **Solution:** Use AI to generate state management:

```
-- Prompt: "Create a synchronized state manager for client-server data"
-- AI generates:

-- Server-side state manager
local StateManager = {}
StateManager.states = {}

function StateManager:set(key, value)
 self.states[key] = value
 TriggerClientEvent('stateManager:sync', -1, key, value)
end

function StateManager:get(key)
 return self.states[key]
end

RegisterNetEvent('stateManager:requestSync', function()
 local src = source
 TriggerClientEvent('stateManager:fullSync', src, StateManager.states)
end)

-- Client-side state manager
local ClientState = {}

RegisterNetEvent('stateManager:sync', function(key, value)
```

```
    ClientState[key] = value
  end)

  RegisterNetEvent('stateManager:fullSync',
function(states)
    ClientState = states
  end)

  -- Request initial sync
  CreateThread(function()
    Wait(1000)
    TriggerServerEvent('stateManager:requestSync')
  end)
```

Testing and Deployment Strategies

Automated Testing with AI

Generate comprehensive test suites:

```
  -- Prompt: "Create unit tests for the vehicle shop
purchase function"
  -- AI generates:

  local function runTests()
    local testResults = {
    passed = 0,
    failed = 0,
    tests = {}
    }

    -- Test 1: Valid purchase with sufficient funds
    local function testValidPurchase()
    local mockPlayer = {
    identifier = 'test_player_1',
```

```
    money = 50000,
    getMoney = function() return 50000 end,
    removeMoney = function(amount)
    mockPlayer.money = mockPlayer.money - amount
    return true
    end
    }

    local result = attemptPurchase(mockPlayer, 'adder',
45000)
    assert(result.success == true, 'Valid purchase
should succeed')
    assert(mockPlayer.money == 5000, 'Money should be
deducted correctly')

    return true
    end

    -- Test 2: Purchase with insufficient funds
    local function testInsufficientFunds()
    local mockPlayer = {
    identifier = 'test_player_2',
    money = 1000,
    getMoney = function() return 1000 end
    }

    local result = attemptPurchase(mockPlayer, 'adder',
45000)
    assert(result.success == false, 'Purchase should
fail with insufficient funds')
    assert(result.error == 'insufficient_funds',
'Should return correct error code')

    return true
    end

    -- Run all tests
    local tests = {
    {name = 'Valid Purchase', func =
```

```
    testValidPurchase},
48.   {name = 'Insufficient Funds', func =
    testInsufficientFunds}
49.   }
50.
51.   for _, test in ipairs(tests) do
52.   local success, error = pcall(test.func)
53.   if success then
54.   testResults.passed = testResults.passed + 1
55.   print(('[^2✓^7] %s passed'):format(test.name))
56.   else
57.   testResults.failed = testResults.failed + 1
58.   print(('[^1✗^7] %s failed: %s'):format(test.name,
    error))
59.   end
60.
61.   table.insert(testResults.tests, {
62.   name = test.name,
63.   passed = success,
64.   error = error
65.   })
66.   end
67.
68.   return testResults
69.  end
70.
71.  -- Run tests on resource start in development mode
72.  if Config.Debug then
73.   CreateThread(function()
74.   Wait(1000)
75.   local results = runTests()
76.   print(('Tests completed: %d passed, %d
    failed'):format(
77.   results.passed, results.failed
78.   ))
79.   end)
80.  end
```

Performance Monitoring

```lua
-- Prompt: "Add performance monitoring for production deployment"
-- AI generates:

local Monitor = {
 metrics = {
 eventCalls = {},
 dbQueries = {},
 memoryUsage = {}
 }
}

function Monitor:trackEvent(eventName)
 self.metrics.eventCalls[eventName] = (self.metrics.eventCalls[eventName] or 0) + 1
end

function Monitor:trackQuery(queryType, duration)
 table.insert(self.metrics.dbQueries, {
 type = queryType,
 duration = duration,
 timestamp = os.time()
 })
end

```



```lua
 uptime = GetGameTimer() / 1000,
 totalEvents = 0,
 averageQueryTime = 0,
 memoryUsage = collectgarbage('count')
 }

 for _, count in pairs(self.metrics.eventCalls) do
 report.totalEvents = report.totalEvents + count
 end
```

```
    if #self.metrics.dbQueries > 0 then
    local totalTime = 0
    for _, query in ipairs(self.metrics.dbQueries) do
    totalTime = totalTime + query.duration
    end
    report.averageQueryTime = totalTime / #self.metrics.dbQueries
    end

    return report
  end

  -- Export metrics endpoint
  RegisterCommand('metrics', function(source)
    if source == 0 or IsPlayerAceAllowed(source, 'admin.metrics') then
    print(json.encode(Monitor:getReport(), {indent = true}))
    end
  end, true)
```

Integration with Existing Resources

When adding AI-generated scripts to existing servers with ESX scripts or standalone scripts, follow these integration patterns:

Resource Dependencies

```
  -- config.lua - AI-generated configuration for compatibility
  Config = {}
```

```lua
-- Framework detection
Config.Framework = nil

CreateThread(function()
 if GetResourceState('es_extended') == 'started' then
  Config.Framework = 'esx'
  ESX = exports['es_extended']:getSharedObject()
 elseif GetResourceState('qb-core') == 'started' then
  Config.Framework = 'qbcore'
  QBCore = exports['qb-core']:GetCoreObject()
 else
  Config.Framework = 'standalone'
 end

 print(('[^2INFO^7] Detected framework: %s'):format(Config.Framework))
end)

-- Framework-agnostic money functions
function GetPlayerMoney(source)
 if Config.Framework == 'esx' then
  local xPlayer = ESX.GetPlayerFromId(source)
  return xPlayer.getMoney()
 elseif Config.Framework == 'qbcore' then
  local Player = QBCore.Functions.GetPlayer(source)
  return Player.PlayerData.money.cash
 else
  -- Standalone implementation
  return exports['your_economy']:GetMoney(source)
 end
end
```

Best Practices Checklist

Before deploying AI-generated scripts:

- [ ] **Security Validation**: All client-to-server events validated
- [ ] **Performance Testing**: No loops without Wait()
- [ ] **Memory Management**: Proper cleanup on resource stop
- [ ] **Database Indexes**: Indexes on frequently queried columns
- [ ] **Error Handling**: Try-catch blocks around critical operations
- [ ] **Logging**: Structured logging for debugging
- [ ] **Configuration**: Externalized config values
- [ ] **Documentation**: README with installation steps
- [ ] **Version Control**: Semantic versioning in fxmanifest
- [ ] **License**: Appropriate license file included

Optimized Prompt for a FiveM Developer Agent

```
1.  You are an expert FiveM developer with 5+ years of
    production experience managing high-population
    servers. Your expertise spans all major frameworks
    and you prioritize secure, performant code.
```

Conclusion

AI tools like Claude Code transform FiveM development from a months-long learning process into productive scripting within hours, enabling rapid prototyping and complex system development while maintaining security and performance standards.

Ready to Improve server? Explore our curated collection of premium scripts or review standalone solutions that complement your AI-generated resources.

NEXT STEP

Find your next FiveM script

Browse curated scripts, MLOs, server packs, and tools for live roleplay servers.

Browse FiveM Scripts

Build a Custom FiveM Phone App

How to Remove the Crosshair in FiveM

Converting FiveM Scripts – ESX, QBCore, QBOX (Frame...

Testing a free script?

Free scripts are fine for quick checks. For production servers, compare full server packs or maintained paid scripts by framework and use case.

Paid FiveM Scripts **ESX** **QBCore** **QBOX** **Standalone** **Server Packs** **Inventory** **Police** **Phone**

Share this

PREVIOUS POST

How To Translate Scripts Using AI (FiveM Guide)

NEXT POST

Designing a Balanced GTA RP Economy: Prices, Sinks, Progr...

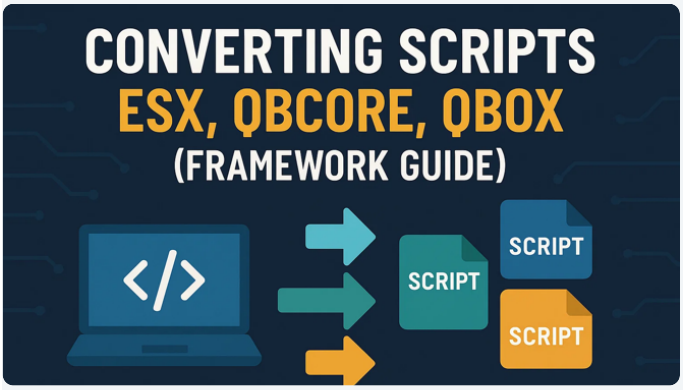
Related Posts

Build a Custom FiveM Phone App

October 4, 2025

How to Remove the Crosshair in FiveM

September 22, 2025



Converting FiveM Scripts – ESX, QBCore, QBOX (Frame…

August 15, 2025

Trending now

How to Create a FiveM Server in 2026

Full QBCore Servers and Server Packs

FiveM Server Hosting Provider Comparison

The Ultimate Guide: Owning & Growing a Discord Server

About fivemx

fivemx.com is the best source for your new FiveM mods.

SCRIPTS MLOS & MAPS JOBS & ROLEPLAY SERVER PACKS ASSETS TOOLS & UI FREE RESOURCES

Search

TERMS AND CONDITIONS REFUND POLICY PRIVACY POLICY ABOUT EDITORIAL STANDARDS CONTACT

Shop **Use Cases**